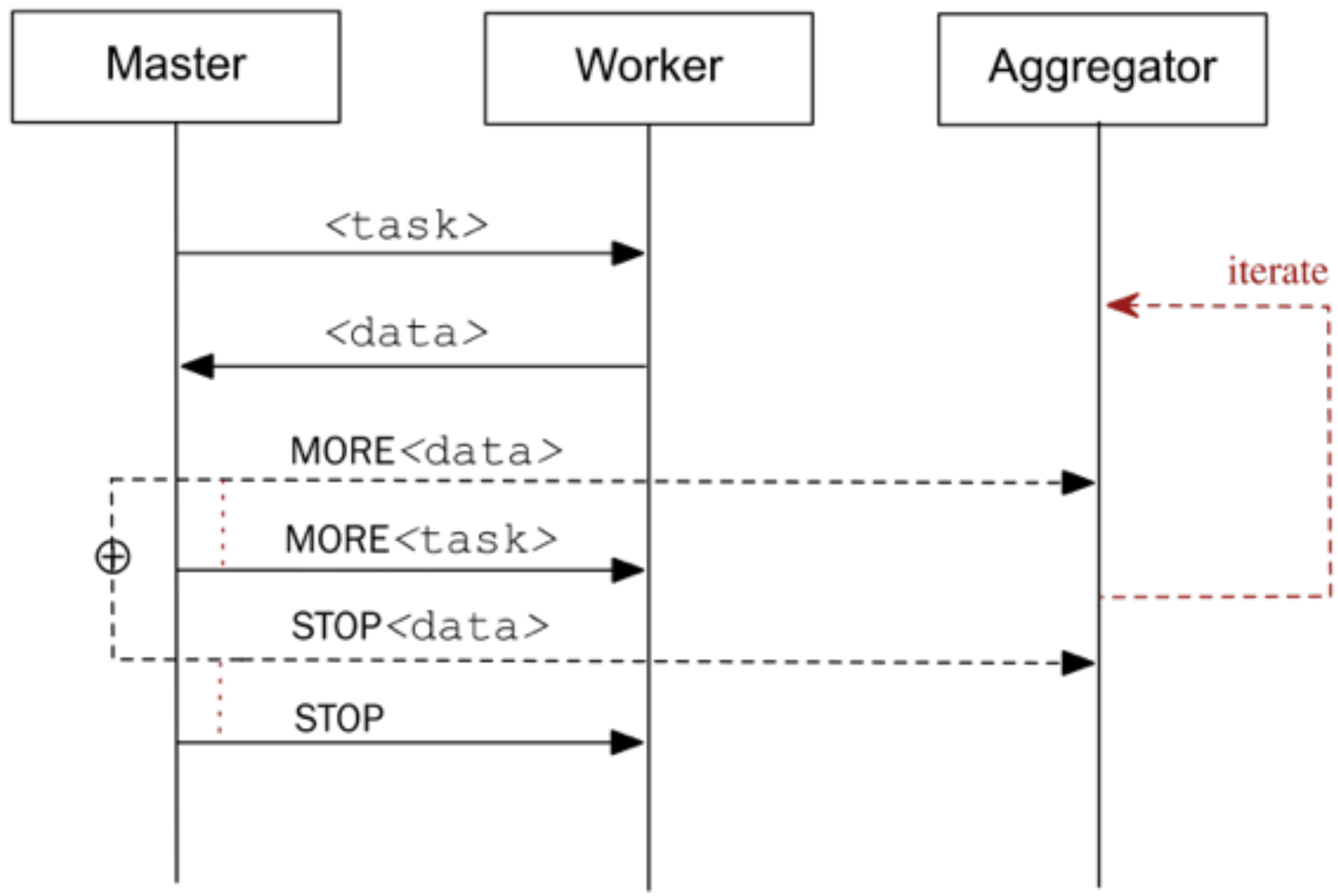
Timed Multiparty Session Types

Laura Bocchi, Weizhen Yang, Nobuko Yoshida

CONCUR 2014

The problem (1)

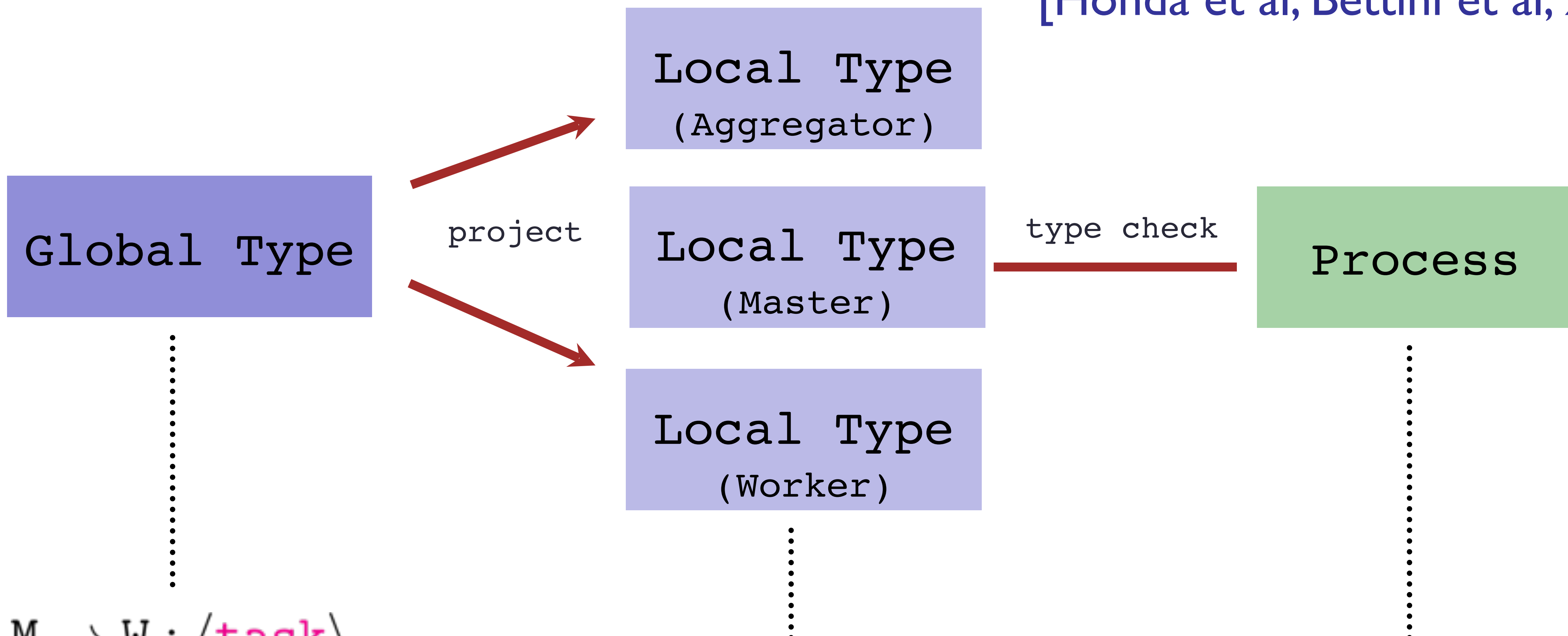
- Independent programs realise ***global tasks*** through network interactions



- Participants need to agree on **protocol**, on **data semantics** …

Multiparty Session Types

[Honda et al, Bettini et al, 2008]



$\mathtt{M} \to \mathtt{W} : \langle \mathtt{task} \rangle.$
$\mathtt{W} \to \mathtt{M} : \langle \mathtt{data} \rangle.$
$\mathtt{M} \to \mathtt{A} : \langle \mathtt{data} \rangle.$
$\mathtt{end}$

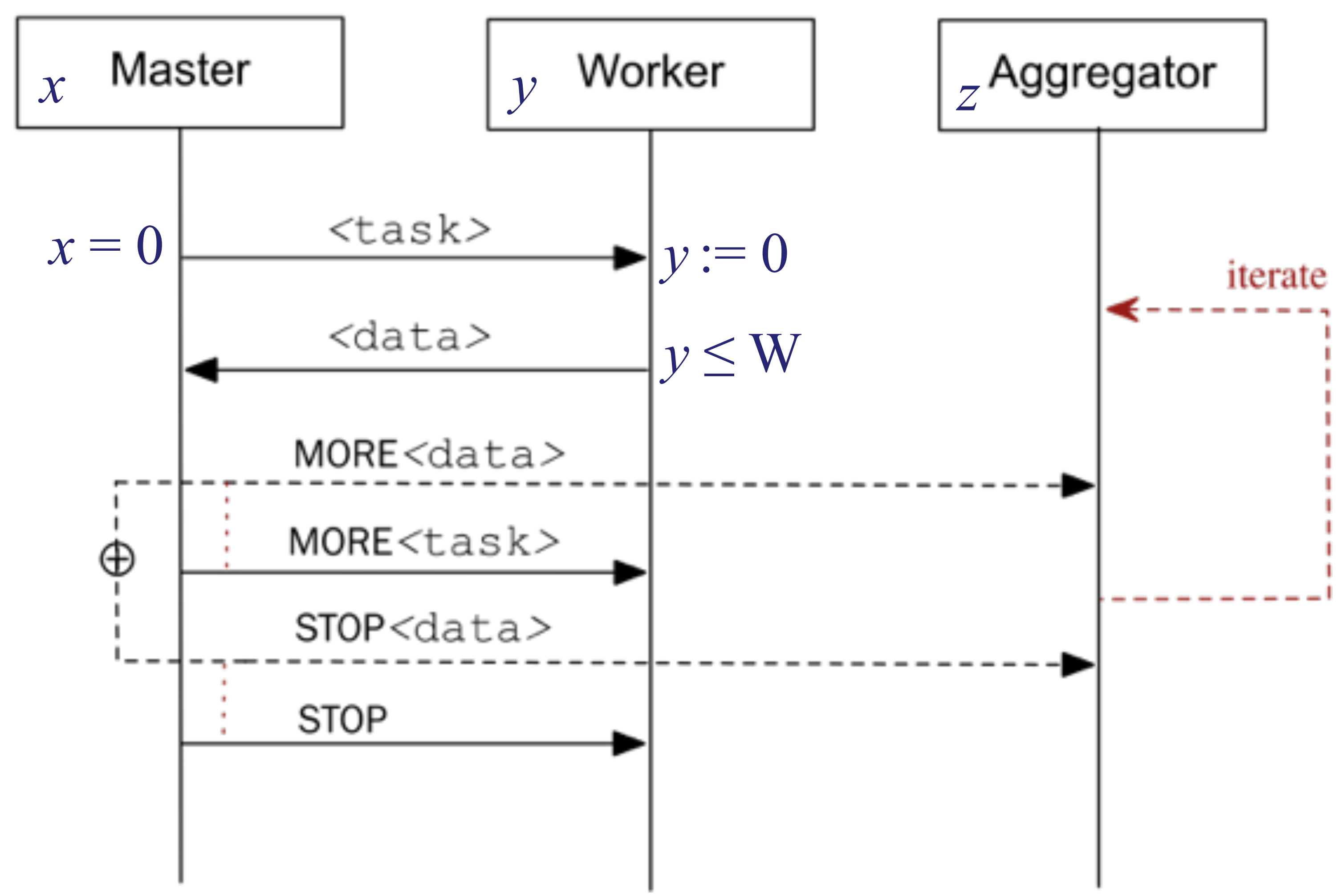
$?\,\mathtt{M}(\mathtt{task}).$
$!\,\mathtt{M}\langle \mathtt{data} \rangle.$
$\mathtt{end}$

$c[\mathtt{M}]?(x).\ \mathtt{if}\ x = \text{“}sample\text{”}$
$\quad \mathtt{then}\ c[\mathtt{M}]!\langle f(x) \rangle$
$\quad \mathtt{else} \ldots$

- **Efficient**, **local** verification of **global** properties (session fidelity & progress)

The problem (2)

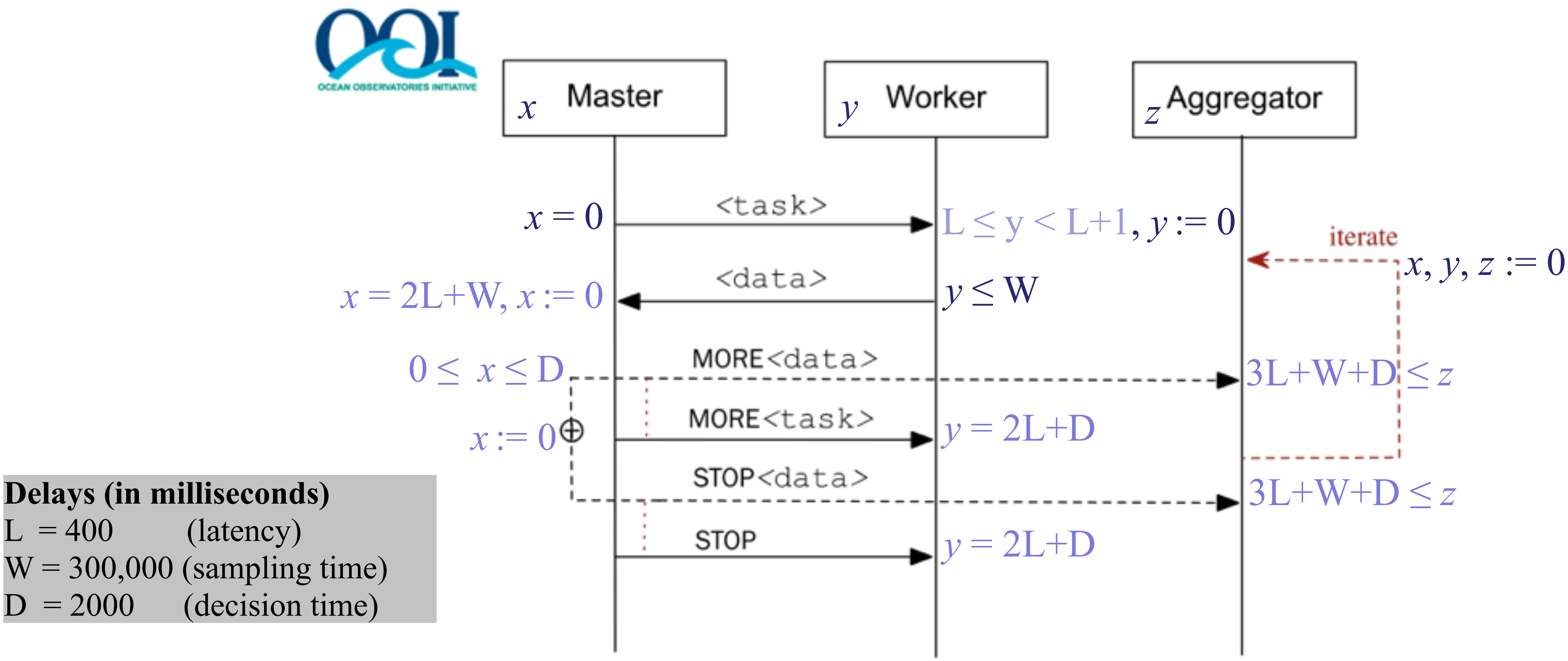
- **Web Services**: *"Reconnect no more than twice every four minutes …"* [Twitter Streaming API]
- **Sensor Networks** (on busy waiting): *"Main sources of energy inefficency in Sensor Networks are collisions and listening on idle channels"* [Ye, Heidemann & Estrin, 2002]
- **Protocol specification**: deadlines, timeouts, repeated constraints (**resets**), …



Delays (in milliseconds)
L = 400 (latency)
W = 300,000 (sampling time)
D = 2000 (decision time)

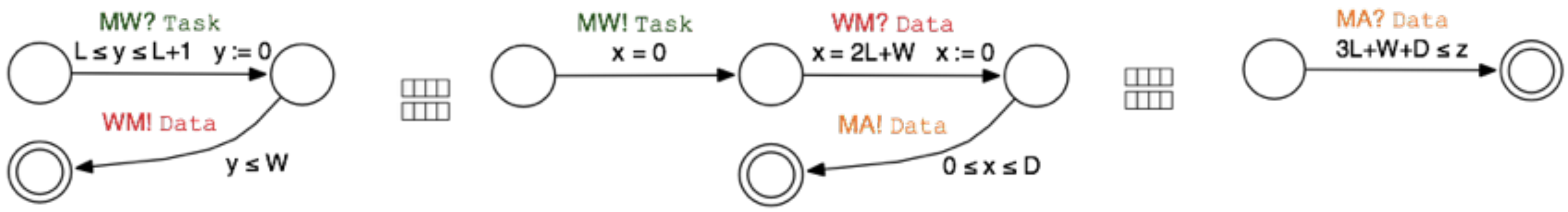
The problem (2)

- **Web Services**: *"Reconnect no more than twice every four minutes …"* [Twitter Streaming API]
- **Sensor Networks** (on busy waiting): *"Main sources of energy ineffficency in Sensor Networks are collisions and listening on idle channels"* [Ye, Heidemann & Estrin, 2002]
- **Protocol specification**: deadlines, timeouts, repeated constraints (**resets**), …



Communicating Timed Automata (CTAs)

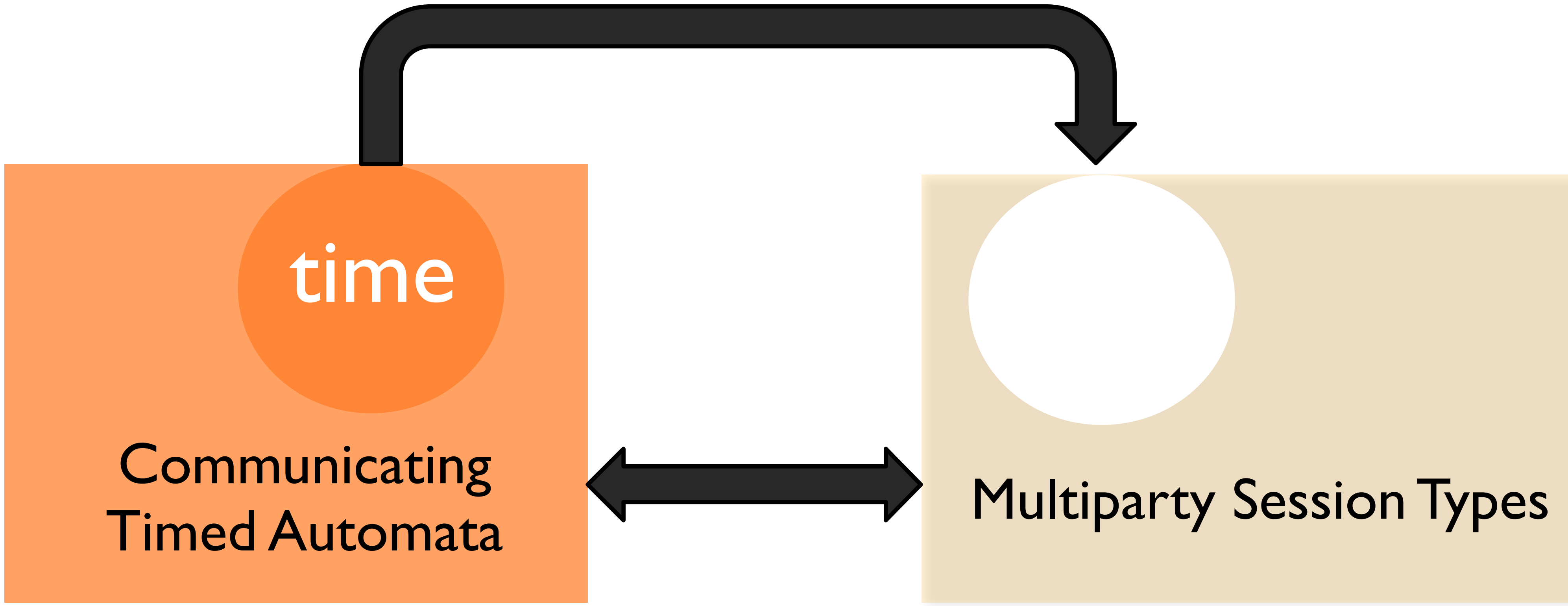
- **Timed automata** [Alur & Dill '94]: set of clocks, guarded transitions, resets



- **CTA** [Krcal & Yi '06]: network of timed automata asynchronously communicating on *unbounded channels* and synchronizing over *time actions*

- In general it is hard to verify properties such as **reachability**
  - upper bound on channels [S. Akshay et al. FSTTCS'94]
  - some topologies e.g., polyforests [Clemente et al. FOSSACS'13]

Overview



- Verification of real-time interactions with Multiparty Session Types
  - ***time-error freedom***: interactions are punctual
  - ***time-progress***: a deadlock state is not ***reachable*** and time can diverge
- Correspondence between global types and Communicating Finite States Machines (CFSMs) [Denielou & Yoshida, ICALP'13]
- Decidable conditions for ***progress*** and ***liveness*** for CTAs

Timed Global Types

$$\mathtt{G} \ ::= \ \mathtt{p} \to \mathtt{q} : \{l_i\langle S_i\rangle\{A_i, A'_i\}.\mathtt{G}_i\}_{i\in I} \ \mid \ \mu\mathtt{t}.\mathtt{G} \ \mid \ \mathtt{t} \ \mid \ \mathtt{end}$$

$$A ::= \delta, \lambda$$

Clock constraint $\delta \ ::= \mathtt{true} \mid x > c \mid x = c \mid \neg\delta \mid \delta_1 \wedge \delta_2$

Resets $\lambda \subseteq \mathcal{X}$ reset $\qquad x, x', y, \ldots \in \mathcal{X}$ clocks

$$
\begin{array}{lllllll}
\mu\mathtt{t}. & \mathtt{M} \to \mathtt{W} : \langle\mathtt{task}\rangle & \{x = 0, & \emptyset, & L \leq y \leq L+1, & y\}. \\
 & \mathtt{W} \to \mathtt{M} : \langle\mathtt{data}\rangle & \{y \leq W, & \emptyset, & x = 2L + W, & x\}. \\
 & \mathtt{M} \to \mathtt{A} : \{\mathsf{MORE}\langle\mathtt{data}\rangle & \{x \leq D, & \emptyset, & z \geq 3L + W + D, & z\}. \\
 & \quad \mathtt{M} \to \mathtt{W} : \mathsf{MORE}\langle\mathtt{task}\rangle & \{x \leq D, & x, & y = 2L + D, & y\}. \\
 & \quad \mathtt{t}\ , & & & & \\
 & \mathsf{STOP}\langle\mathtt{data}\rangle & \{x \leq D, & x, & z \geq 3L + W + D, & \emptyset\}. \\
 & \quad \mathtt{M} \to \mathtt{W} : \mathsf{STOP} & \{x \leq W, & x, & y = 2L + D, & \emptyset\}. \\
 & \quad \mathtt{end} & & & & \\
 & \} & & & &
\end{array}
$$

Transitions & satisfiability of ready actions

$x = 0,\ y = 0$ | $\mathtt{p} \to \mathtt{q} : \langle \mathtt{int} \rangle \quad \{1 \le x \le 10,\quad \emptyset,\quad y > 10,\quad \emptyset\}.\ \mathtt{end}$

Specified semantics:

$\xrightarrow{\mathtt{pq}!\langle \mathtt{int} \rangle}$

$\xrightarrow{15}$

$\xrightarrow{1.99} \quad \xrightarrow{\mathtt{pq}!\langle \mathtt{int} \rangle}$

Transitions & satisfiability of ready actions

$x = 0,\ y = 0$ | $\mathtt{p} \to \mathtt{q} : \langle \mathtt{int} \rangle \quad \{1 \leq x \leq 10,\quad \emptyset,\quad y > 10,\quad \emptyset\}.\ \mathtt{end}$

Specified semantics:

✘ $\xrightarrow{\mathtt{pq}!\langle \mathtt{int} \rangle}$

Communication actions at times that violate the constraints are not allowed

$\xrightarrow{15}$

$\xrightarrow{1.99} \xrightarrow{\mathtt{pq}!\langle \mathtt{int} \rangle}$

Transitions & satisfiability of ready actions

$x = 0,\ y = 0$ | $\mathtt{p} \to \mathtt{q} : \langle \mathtt{int} \rangle \quad \{1 \leq x \leq 10,\ \ \emptyset,\ \ y > 10,\ \ \emptyset\}.\ \mathtt{end}$

Specified semantics:

✘ $\xrightarrow{\mathtt{pq}!\langle \mathtt{int} \rangle}$

Communication actions at times that violate the constraints are not allowed

✘ $\xrightarrow{15}$

Time actions that make constraints of some **ready** action unsatisfiable are not allowed

$\xrightarrow{1.99}\ \xrightarrow{\mathtt{pq}!\langle \mathtt{int} \rangle}$

Transitions & satisfiability of ready actions

$x = 0,\ y = 0$ | $\mathtt{p} \to \mathtt{q} : \langle \mathtt{int} \rangle \quad \{1 \le x \le 10,\quad \emptyset,\quad y > 10,\quad \emptyset\}.\ \mathtt{end}$

Specified semantics:

✘ $\xrightarrow{\mathtt{pq}!\langle \mathtt{int} \rangle}$

Communication actions at times that violate the constraints are not allowed

✘ $\xrightarrow{15}$

Time actions that make constraints of some **ready** action unsatisfiable are not allowed

✔ $\xrightarrow{1.99} \quad \xrightarrow{\mathtt{pq}!\langle \mathtt{int} \rangle}$

A (very) simple timed calculus

- **Calculi with time:** inspired by CTAs [Saeedloei et al '13], with timeouts [Laneve et al. 05, Berger et al. '07, Lopez et al. '12], for SOC [Lapadula et al. '07] …
- We use a *very* simple calculus

$P :=$	$\overline{u}[\mathtt{n}](y).P$	Request	$\mid$	$P \mid Q$	Parallel
$\mid$	$u[\mathtt{i}](y).P$	Accept	$\mid$	0	Inaction
$\mid$	$c[\mathtt{p}] \triangleleft l\langle e\rangle; P$	Select	$\mid$	$\mu X.P$	Recursion
$\mid$	$c[\mathtt{p}] \triangleright \{l_i(z_i).P_i\}_{i\in I}$	Branching	$\mid$	X	Variable
$\mid$	$\mathtt{delay}(t).P$	Delay	$\mid$	$(\nu a)P$	Hide Shared
$\mid$	$\mathtt{if}\ e\ \mathtt{then}\ P\ \mathtt{else}\ Q$	Conditional			

Process P

$$\mathtt{delay}(400).\ c[\mathtt{M}] \triangleright (x).$$
$$\mathtt{delay}(200000).\ c[\mathtt{M}] \triangleleft \langle f(x)\rangle;$$
$$0$$

Type T

$$\mathtt{M}\ \&\ (\mathtt{task})\{400 \le y < 401,\ y := 0\}.$$
$$\mathtt{M} \oplus \langle\mathtt{command}\rangle\{y \le 300000,\ \emptyset\}.$$
$$\mathtt{end}$$

Type-safety

- We give a type system for timed processes based on judgments:

$$\Gamma \vdash P \triangleright \Delta \qquad\qquad \Delta ::= \emptyset \mid \Delta, c : (\nu, \mathtt{T})$$

$$\frac{\Gamma \vdash P \triangleright \{c_i : (\nu_i + t, \mathtt{T}_i)\}_{i \in I}}{\Gamma \vdash \mathtt{delay}(t).P \triangleright \{c_i : (\nu_i, \mathtt{T}_i)\}_{i \in I}} \quad \text{[Delay]}$$

$$\frac{j \in I \quad \Gamma \vdash e : S_j \quad \nu \models \delta_j \quad \Gamma \vdash P \triangleright \Delta, c : ([\lambda_j \mapsto 0]\nu, \mathtt{T}_j)}{\Gamma \vdash c[\mathtt{p}] \triangleleft l_j\langle e\rangle; P \triangleright \Delta, c : (\nu, \mathtt{p} \oplus \{l_i\langle S_i\rangle\{\delta_i, \lambda_i\}.\mathtt{T}_i\}_{i \in I})} \quad \text{[Select]}$$

Theorem (Time-error freedom) If $\Gamma \vdash P \triangleright \Delta$, and $P \longrightarrow^* P'$ then $P' \neq \mathtt{error}$

An action has been executed at a time that violates the specification

Time progress

- Time-progress (for well-typed timed processes):
  - each reachable state is not a deadlock state (it is *final* or it can *reduce*)
  - time can diverge (the only possible way forward must not be *Zeno*)

- Well-typedness of timed processes does **not** guarantee progress in general
- We give two **decidable** sufficient conditions:
  - Feasibility
  - Wait-freedom

Feasbility

A constraint in a timed global type may not be satisfiable

✘ $\mathtt{p} \to \mathtt{q} : \langle \mathtt{Int} \rangle \{x > 3,\ \emptyset,\ y = 4,\ \emptyset\}.\mathtt{end}$

✔ $\mathtt{p} \to \mathtt{q} : \langle \mathtt{Int} \rangle \{x > 3 \wedge x \leq 4,\ \emptyset,\ y = 4,\ \emptyset\}.\mathtt{end}$

✔ $\mathtt{p} \to \mathtt{q} : \langle \mathtt{Int} \rangle \{x > 3,\ \emptyset,\ y \geq 4,\ \emptyset\}.\mathtt{end}$

Feasibility: *for each partial execution allowed by a specification there is a correct complete one* [Apt, Francez & Katz, POPL'87]

Wait-freedom

Some *well-typed* distributed implementation of *feasible* timed types may lead to inconsistent views of the timing of actions by different participants.

The receiver may not find the message ready when reading the channel

$$\mathtt{p} \to \mathtt{q} : \langle \mathtt{Int} \rangle \{ x_{\mathtt{p}} < 3 \vee x_{\mathtt{p}} > 3,\ x_{\mathtt{q}} < 3 \vee x_{\mathtt{q}} > 3 \}.\mathtt{G}$$

$$P = \mathtt{delay}(6).\ c[\mathtt{q}] \triangleleft \langle 10 \rangle ; \mathtt{G} \downarrow \mathtt{p}$$
$$Q = c[\mathtt{p}] \triangleright (x).\mathtt{G} \downarrow \mathtt{q}$$

$$P \mid Q \not\longrightarrow$$

P can send the message only after 6 time units elapse

Q needs to receive now … but the message is not ready!

Wait-freedom: *The constraint of each receive action must not admit, as a solution, a time which is earlier than some solution of the corresponding send action.*

Progress for t-MPSTs

- Well-typed processes of **feasible** and **wait-free** MPSTs enjoy global progress *(if their untimed counter-part does so)*
  - processes in single sessions (e.g., [Honda et al. POPL'08])
  - processes in interleaved sessions (e.g., [Bettini et al. CONCUR'08]) where session initiations do not occur after delays

✘ $\texttt{delay}(6).\ \overline{a}[1].P \mid a[1].Q$

✔ $\overline{a}[1].\texttt{delay}(6).P \mid a[1].Q$

Correspondence with CTAs

- In the untimed setting [Denielou & Yoshida, ICALP'13] gives correspondence between global types, and basic and multiparty compatible CFSMs

- In the timed setting we give correspondence between timed global types, and basic and multiparty compatible CTAs with a *specified semantics*

- A **CTA** that is basic, multiparty compatible, with specified semantics and corresponds to a **feasible** timed global type ensures
  - **progress**: each *reachable* state is *non-deadlock* and allows *time divergence*
  - **liveness**: a final state can be reached from all reachable states

Conclusion & future work

MPSTs

- *Timed Multiparty Session Types* & typing system for *timed processes* ensuring *time-error freedom*
- *Progress* of well-typed implementations of feasible and wait-free types
- Feasibility and wait-freedom decidable for *infinite satisfiable* timed global types
- Implementation in [Neykova et al. BEAT'14]
- *Future work*: can we extend timed MPSTs with parallel operation?

CTAs

- Correspondence between CTAs and MPSTs
- A class of CTAs enjoying progress and liveness (based on *feasibility*)
- We used CTAs as *types*. Progress of CTAs only requires feasibility
- Recently, we convert from CTAs to timed MPSTs

Questions?